

How Enqpy™ Proves It Can't Be Read

A Plain Language Guide to the World's First Cipher Formally Proved Unreadable — Not Just Hard to Break

NQP LLC • Manassas, Virginia • 2026 • Rev 5.0 (June 2026)
Written for curious minds — no math class required

What This Guide Covers

This guide walks you through the complete proof that Enqpy™ is provably unreadable — using everyday examples, analogies, and plain language. No equations, no jargon. By the end you'll understand exactly why no computer — including quantum computers — can ever crack it.

1. What Does “Absolutely Secure” Actually Mean?

Before we explain the proof, we need to be really precise about what “secure” means. Because there are two completely different ways to say a secret code is secure — and only one of them is absolutely secure, no matter how much computing power an attacker has.

Computationally Hard (Not Absolutely Secure) Most ciphers today — including every government-approved standard — are designed to be very hard to crack. Think of it like a giant combination lock with a trillion trillion possible combinations. Nobody has time to try them all. But “nobody has time” is different from “impossible.” If computers get faster, the lock gets easier. One NIST post-quantum finalist called SIKE was broken in 62 minutes on a 2013 laptop using a clever shortcut. The lock wasn't picked by brute force — someone found the door.	Information-Theoretically Impossible (Absolutely Secure) Enqpy™ claims something completely different: no amount of computing power can ever break it, because breaking it would require information that literally does not exist in the ciphertext. The problem isn't that the computation is hard. The problem is that the math guarantees the secret simply isn't there in the ciphertext to recover. Claude Shannon — the father of information theory — defined exactly what this means in 1949.
--	---

□ ANALOGY

Imagine a detective trying to identify a murderer. If there were 1,000 possible suspects, a brilliant detective with enough time could work through clues and narrow it down. That's computational hardness — hard but solvable given enough work. Now imagine a detective trying to identify a murderer when the crime was committed in a perfectly sealed room with no evidence of any kind — and there are genuinely 2 people who could each have done it with equal probability, and nothing that exists anywhere can distinguish between them. That's information-theoretic security. It's not that solving it is hard. It's that the answer doesn't exist in any unique form.

□ KEY POINT

Shannon's famous result: A cipher is absolutely secure — in the information-theoretic sense — if and only if, no matter how much ciphertext an enemy collects, they are always left with at least 2 possible keys that are equally consistent with everything they've seen. This is called “key equivocation that never goes to zero.” Enqpy™ is the first practical cipher in history to prove this formally.

2. The Big Idea: Answers That Permanently Disappear

Here's the core idea behind Enqpy™, stated as simply as possible:

□ KEY POINT

Enqpy™ uses a math operation where the answer permanently forgets how it was created. No matter how hard you try to work backward from the answer, you will always find at least 2 different ways that could have produced it — and nothing in that output can tell you which one was actually used.

The specific math operation is called MOD16 addition. Let's understand exactly what that means.

2.1 MOD16 Addition: The Core of Everything

Normal addition: $7 + 5 = 12$. Simple. You can work backward: if the answer is 12 and one number is 7, the other must be 5.

MOD16 addition works differently. It's like arithmetic on a clock that only has 16 positions (labeled 0 through 15 instead of 1 through 12).

□ ANALOGY

Imagine a clock with only 16 positions: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15. When you go past 15, you wrap back around to 0. So $14 + 5$ on this clock = 3 (because $14 + 5 = 19$,

and $19 - 16 = 3$). Now here's the key thing: if someone tells you the answer is 3, can you figure out which numbers were added? $14 + 5 = 3 \equiv \text{Yes}$. But also $0 + 3 = 3 \equiv \text{Yes}$. And $1 + 2 = 3 \equiv \text{Yes}$. And $7 + 12 = 19 = 3 \pmod{16} \equiv \text{Yes}$. And $8 + 11 = 19 = 3 \pmod{16} \equiv \text{Yes}$. There are exactly 16 pairs that all give the answer 3.

□ THE MATH

Lemma 1 (the first proof result): For any output value r , there are exactly 16 different input pairs (a, b) where $a + b \equiv r \pmod{16}$. This is true for every possible output value — not just 3. Every answer always has exactly 16 ways it could have been created. This is a pure mathematical fact about the structure of arithmetic, with no exceptions.

□ EXAMPLE

Let's check this for output = 0. What pairs (a, b) satisfy $a + b \equiv 0 \pmod{16}$? $0+0=0$, $1+15=16=0$, $2+14=16=0$, $3+13=16=0$, $4+12=16=0$, $5+11=16=0$, $6+10=16=0$, $7+9=16=0$, $8+8=16=0$, $9+7=16=0$, $10+6=16=0$, $11+5=16=0$, $12+4=16=0$, $13+3=16=0$, $14+2=16=0$, $15+1=16=0$. That's exactly 16 pairs. Every other output value also has exactly 16 pairs. Always.

□ KEY POINT

This is Lemma 1. It's the mathematical foundation that everything else rests on. It's not a guess or an assumption — it's a provable algebraic fact about MOD16 arithmetic. You can verify it yourself by checking every pair.

3. The Key Machine (PDAF): How Enqpy™ Uses MOD16

Enqpy™ doesn't just do one MOD16 addition. It runs the operation thousands of times in a clever connected pattern called PDAF (you don't need to remember that name — just think of it as the Key Machine).

3.1 How the Key Machine Works

When Enqpy™ wants to encrypt a message, it needs to create a secret per-message key. It does this by feeding your master secret key into the Key Machine. The machine runs thousands of MOD16 additions in a specific pattern, and the output becomes the session key used to scramble the message.

□ ANALOGY

Think of the Key Machine like a blender with a very specific recipe. You pour in your secret ingredients (the master key). The blender runs a very specific sequence of mixing steps — all based on that 16-pair MOD16 property. What comes out is a smoothie (the session key). Here's the critical thing: given only the smoothie, you cannot figure out exactly what the

original ingredients were. Because of that 16-pairs property, there are always multiple ingredient combinations that would have produced the exact same smoothie.

3.2 The Connection Pattern and What It Creates

The Key Machine doesn't just run random MOD16 additions. It runs them in a specific interconnected pattern. Each output position depends on exactly two positions in the input, combined with a displacement value. This creates a web of connections that can be drawn as a graph — like a network diagram where each node is a key position and each line is a constraint.

□ ANALOGY

Imagine a network of 64 people standing in a circle. Each person's "code value" must satisfy an equation involving themselves and one neighbor (where the neighbor is selected by the displacement pattern). The equations form a web. Now suppose someone knows all the resulting outputs. Can they figure out all the original values? Sometimes yes, mostly no — because multiple sets of original values can produce the same outputs. The key question is: how many sets? That's exactly what Theorem 1 answers.

3.3 Cycles: The Secret of Why Ambiguity Persists

The connection pattern creates loops — called cycles — in the network. This is where it gets interesting. The behavior of these cycles determines exactly how many different keys could have produced any given output.

□ ANALOGY

Think of the network as a game of telephone around a circular table. If you have 4 people in a circle (an even cycle), and each person's message depends on their neighbor, you can have 16 different starting values at any seat and still end up with the same final pattern. The freedom is unrestricted. But if you have 3 people in a circle (an odd cycle), something special happens. When you trace the chain of dependencies all the way around and come back to the start, the equation that must be satisfied is: $2 \times (\text{starting value}) = \text{some target} \pmod{16}$. And on our 16-position clock, multiplying by 2 cuts the number of solutions in half — from 16 down to exactly 2.

□ THE MATH

Theorem 1 (the second proof result): For any output that the Key Machine can actually produce, the number of different input keys that could produce that exact output is always at least 2. • If the connection graph has no odd-length loops: at least 16 possible input keys. • If the connection graph has at least one odd-length loop: exactly 2 possible input keys. This is the absolute minimum. In the worst case, there are always 2. Often there are many more.

□ KEY POINT

The magic: The 2 remaining possible keys in the worst case are not ‘close’ to each other or obviously related in a usable way. They are mathematically equivalent from an attacker’s perspective — both are perfectly consistent with every observation the attacker can ever make. No amount of computation, no clever algorithm, no quantum computer can distinguish between them, because the ciphertext contains no information that could make the distinction.

3.4 Why Exactly 2? (The +8 Trick)

The two surviving keys always have a specific mathematical relationship: one is the true key, and the other is the true key with 8 added to every nibble (the +8 global shift). This is because adding 8 twice on a 16-position clock brings you back exactly to where you started ($8 + 8 = 16 \equiv 0 \pmod{16}$). So the Key Machine always produces identical output for a key and for “key + 8 everywhere.”

□ ANALOGY

Imagine a clock face where 12 and 6 are perfectly symmetric — every equation involving the clock hand works out identically whether you start at 12 or at 6. You cannot tell whether the clock started at 12 or at 6, because every measurement you take is identical for both. That’s the +8 shift. The key and the key-plus-8 are the two permanent survivors.

□ EXAMPLE

True key: [3, 11, 7, 14, 2, ...] (64 numbers) Alternate key: [11, 3, 15, 6, 10, ...] (each number is 8 higher, wrapping around) Both produce identical output from the Key Machine for every possible input. Every observation an attacker makes is consistent with both. There is no experiment, no data collection, no computation that can determine which is the actual key. The distinction simply does not exist in the ciphertext.

4. The Full Cipher: Five Phases

Now we know the Key Machine can’t be inverted to a unique answer. But a cipher needs to do more than just generate a session key — it needs to actually encrypt your messages. Enqpy™ runs five phases every time you send a message.

Phase	Name	What Happens	Why It Matters
1	Fresh Nonce	A random public number is generated and mixed with a counter. This becomes the “eff_or” — the unique ID for this message.	Ensures every message uses a completely different session key, even with the same master keys.
2	Session Key Generation	The Key Machine runs twice — once with EK as input, once with QK as	This is where the irreversibility property from Theorem 1 kicks in.

		input — producing the session keys VKC and OKC.	
3	Selection (W generation)	A per-unit keystream W is derived from VKC and OKC via the canonical Case-1 selection rule.	Creates the actual keystream bytes that will be XOR'd with the message.
4	Encrypt / Decrypt	Each byte of plaintext is XOR'd with the corresponding byte of W to produce ciphertext (or vice versa).	XOR is perfectly reversible when you have W — but you need the exact W, which requires the master keys.
5	Key Update	VKC and OKC are updated via the Key Machine after each W cycle.	Provides in-session key evolution: keystream material is never reused across cycle boundaries within a message. Cross-session isolation comes from the unique nonce (eff_or) for each message, not from Phase 5.

Two master keys are always used: EK (Encryption Key) and QK (Query Key). They must be different from each other. This two-key design means that compromising one key does not reveal the other.

4.1 The Nonce: Why Every Message Is Fresh

The “nonce” (Phase 1) is a random number that gets mixed into the key derivation for each message. Because the nonce is different every time, the session keys VKC and OKC are different for every message, even though the master keys EK and QK stay the same.

□ **ANALOGY**

Your master keys EK and QK are like a chef’s unchanging knowledge and skill. The nonce is like the specific fresh ingredients available today. Even with the same chef and the same recipe, each day’s meal tastes different because the ingredients are different. An attacker who somehow figured out today’s session key still has no idea what yesterday’s or tomorrow’s session key was, because each day used a completely different nonce.

5. The Big Theorem: Why Collecting More Messages Never Helps

Here’s the heart of the proof. The attacker is doing everything possible: they’re capturing every encrypted message you send — every byte of ciphertext, plus the public nonce that travels with it. They never see your plaintext; they only have what crosses the wire. They’re building up a massive pile of ciphertext, hoping that eventually the pattern will give away the master keys — or the messages.

Theorem 2 proves this can never work. Here's why, step by step.

5.1 What the Attacker Actually Has

The attacker captures the ciphertext C and the public nonce that travels with it. That is the whole of what they have. They never see your plaintext M — if they could, the message would not be a secret to begin with. And without the plaintext they cannot compute the keystream W either, because that would take $W = C \oplus M$. So they are holding scrambled bytes and a public number, and the keystream that links the two stays hidden from them.

Their goal, then, is to work from the ciphertext alone back to either the master keys or the original message. The rest of this section shows why neither is reachable — and it fails on two separate fronts.

□ ANALOGY

Think of four different chefs, each working from a different secret recipe, who nonetheless turn out a byte-for-byte identical smoothie every single time — same flavor, same color, same thickness, whatever fruit you hand them. You can taste a thousand of their smoothies and still never say which chef made any one of them, because the smoothies are genuinely identical. Those four chefs are the four master keys Enqpy™ can never tell apart. The ciphertext is the smoothie.

5.2 Why More Messages Don't Help

Every message uses a different nonce, so every message is scrambled with a different keystream. You might think that capturing enough of them — a billion fresh clues — would eventually triangulate the master keys. It never does, and here is the structural reason:

- There are always four master-key pairs that produce byte-for-byte identical ciphertext: the true pair (EK, QK) and its three +8 relatives — $(EK+8, QK)$, $(EK, QK+8)$, and $(EK+8, QK+8)$.
- Every one of those four pairs reproduces every ciphertext the attacker has ever captured — not approximately, but exactly, down to the byte.
- So no ciphertext, and no quantity of ciphertext, can ever separate them. The captured stream is identical for all four.
- The attacker is left holding four equally valid keys with no way to choose between them — and that stays true no matter how many messages they collect.

□ THE MATH

Theorem 2 (the main result): No matter how much ciphertext an attacker captures — even infinitely much — the four +8-coset master-key pairs stay permanently indistinguishable: $\{EK, EK+8\} \times \{QK, QK+8\}$. The key equivocation $H(EK, QK \mid \text{all ciphertext}) \geq \log_2 4 = 2$ bits — a

floor that never goes to zero. (Exactly four — exactly 2 bits — is the tightness that holds under known plaintext; from ciphertext alone the floor can only be larger.) Enqpy™ satisfies Shannon's Ideal System definition.

5.3 Why the Floor Stabilizes (Doesn't Creep Downward)

A skeptic might ask: maybe the equivocation starts at many bits and slowly decreases toward 2 bits, approaching it asymptotically but never quite reaching it — does that mean it's still weakening over time?

The answer is no. The equivocation doesn't slowly drift down. It snaps to its floor value very quickly (after roughly as many messages as there are nibbles in the key — 64 for the HIGH security profile) and then stays there permanently. After that stabilization point, every additional message collected adds zero additional information about the master keys. The floor is locked in, not approached.

□ EXAMPLE

HIGH profile: 256-bit keys, $n = 64$ nibbles. After about 64 messages: equivocation has completely stabilized. After 1,000 messages: equivocation = same floor (2 bits). After 1 million messages: equivocation = same floor. After all the messages ever sent since the internet began: same floor. With every quantum computer ever built running simultaneously: same floor. The floor doesn't leak. It's structural.

5.4 The Second Axis: Your Message Stays Secret Too

Here's something even more surprising — and something established as new proof work in 2026. The key-equivocation result (Sections 5.1–5.3) proves the attacker can never identify the master keys from the ciphertext. There is an independent second result: from that same ciphertext, they can never determine the original message either. Not the key, not the message — neither can be recovered from what crosses the wire.

□ KEY POINT

Not only are the master keys permanently ambiguous — the original message is permanently ambiguous too. For any ciphertext the attacker observes, at least 2^{128} completely different messages are consistent with it. That's a number with 39 digits. The attacker can never reduce it to a unique answer, no matter what they know or compute.

Here's why. When Enqpy™ encrypts a message, it XORs each byte with a keystream byte. The keystream comes from the Key Machine running on your master keys and the public nonce. But because of that 16-pairs property, the Key Machine can produce many different keystreams — not just one — for different key choices. Each different keystream, when XOR'd with the observed ciphertext, gives back a completely different “plausible” message. There are at least 2^{128} such plausible messages — and all of them are exactly equally probable from the attacker's perspective.

□ ANALOGY

Imagine receiving a wax impression of an unknown object. You know an original object was pressed into the wax to create it. But the pressing process was designed so that at least 2^{128} completely different original objects would produce this exact impression. A sphere could have made it. A cube could have made it. A tiny sculpture of the Eiffel Tower could have made it. You cannot determine which object was pressed just by examining the wax — because the mathematics guarantees that an astronomically large number of objects all produce the exact same result.

□ THE MATH

Theorem 3 (Plaintext Equivocation Theorem, proved 2026 — closed): For any ciphertext CT and any public nonce OR, the number of different messages consistent with (CT, OR) is at least 2^{128} for HIGH profile (256-bit keys). This result is closed: all consistent messages are exactly equally probable under a uniform key prior — the posterior is uniform over the full consistent set, giving $H = H^\infty \geq 128$ bits, and the attacker has no basis to prefer one over another. This result is proved algebraically using the same dependency graph analysis as the key equivocation result, applied to the image structure of the Key Machine rather than its preimage structure. The 2^{128} support floor and the uniform posterior are both proved unconditionally.

□ KEY POINT

This result answers a philosophical objection sometimes raised against the key equivocation result: “Your four equivalent key pairs are really just one key written four ways — so the ambiguity is representational, not real.” Even if that objection were accepted (it isn't), the message-axis result is completely unaffected. The 2^{128} message ambiguity has nothing to do with how keys are counted. It is a direct property of the cipher's output structure. The adversary cannot determine the message regardless of how they count or normalize the keys. The two results together satisfy Shannon's core non-uniqueness criterion: the adversary can pin down neither the key nor the message to a unique answer, on two independent mathematical axes simultaneously.

5.5 Is Enqpy™ “Weaker” Than a One-Time Pad? The Deterministic View

Shannon labeled his Ideal System “weaker” than Perfect Secrecy (the One-Time Pad) back in 1949 — and mathematically, within his original framework, that designation is correct. Perfect Secrecy leaves the adversary facing the complete set of all possible messages. An Ideal System leaves them facing a proper subset — still permanently ambiguous, but smaller. Under a probabilistic measure of how quickly the adversary's odds improve, smaller is technically worse. This is the accurate, careful meaning of “weaker.” But here's the thing: that probabilistic measure answers a different question than the one an attacker actually needs to answer in the real world. April 2026 saw a companion note (“A Deterministic Success-Metric View,” NQP LLC) formally examine what happens when you ask the operational question instead.

□ ANALOGY

Imagine two locked doors. Door A is at the end of an infinitely long corridor — you can walk toward it forever but never reach it. Door B is at the end of a shorter corridor, but that corridor loops back on itself: you walk, and walk, and the door stays permanently out of reach. Shannon's framework measures the rate at which you approach the door. Door A (Perfect Secrecy) is technically "harder to approach" than Door B (Ideal System). But the operational question is: can you ever reach the door? For both doors, the answer is the same: no. The destination is permanently unreachable. When the destination is permanently unreachable, the rate of approach doesn't change the outcome.

The companion note introduces a binary adversarial outcome function called the Deterministic Success Metric S . The metric asks one question: can the adversary determine the answer with certainty? $S = 1$ means yes. $S = 0$ means no. That's it. No measurement of how many alternatives remain, no calculation of how probable each one is — just: did the adversary succeed with certainty?

□ KEY POINT

The Deterministic Success Metric S asks: can the adversary succeed? $S = 1$ if the adversary can uniquely determine the plaintext with certainty. $S = 0$ otherwise — regardless of how many alternatives remain. An important caution: $S = 0$ has two very different forms. **Trivial $S = 0$** occurs when one candidate sits at near-certain probability — the system is operationally broken even though the adversary can't claim perfect certainty. **Substantive $S = 0$** occurs when the impossibility of a unique answer is accompanied by independently proved large residual support — as it is for Enqpy™. For Enqpy™, $S = 0$ is always substantive: the proved support floor $|S(CT, OR)| \geq 2^{128}$ means at least 2^{128} candidates always remain consistent, and no observation can ever force a unique answer. (The posterior is uniform over the full consistent set — $H = H^\infty \geq 128$ bits — so probability concentration is closed.)

Under metric S , both Perfect Secrecy and Enqpy™ produce $S = 0$ for all adversaries, including quantum adversaries, for all quantities of ciphertext. The formal result — Theorem 3.1 in the companion note — is proved by citing Enqpy™'s existing proved results: $H(EK, QK|T^\infty) \geq 2$ bits (key axis) and the support floor $|S(CT, OR)| \geq 2^{128}$ (message axis). Since at least 2^{128} plaintexts remain consistent with any observation, unconditionally, the adversary can never establish a unique answer with certainty. $S = 0$. Shannon's "weaker" designation retains every bit of its mathematical meaning — it just does not change this outcome.

□ KEY POINT — Theorem 3.1: Outcome Equivalence

For Perfect Secrecy (any One-Time Pad): $S = 0$ for all adversaries, all ciphertext. Shannon proved this in 1949. For Enqpy™ (proved Ideal System instantiation): $S = 0$ for all adversaries including quantum, all ciphertext. This follows from the independently proved equivocation bounds. Therefore, under metric S and those bounds, Enqpy™ and Perfect Secrecy share the same adversarial outcome. Shannon's "weaker" designation describes a real mathematical difference. It does not change the answer to the question an adversary actually faces.

There is one more structural observation worth noting. Shannon's framing treats the Ideal System as a degraded One-Time Pad — a system that achieves “less” uncertainty. But when you look at the two systems structurally, something interesting emerges. With a One-Time Pad, key uncertainty and message uncertainty are algebraically coupled: given any ciphertext, if you know the message you can calculate the key exactly, and vice versa ($K = M \oplus C$). The uncertainty is complete, but it lives on a single axis — knowing one side resolves the other. Enqpy™ is different. Its two uncertainty bounds — key equivocation and message equivocation — are algebraically independent. Neither one is derivable from the other. The companion note calls this Uncertainty Duality.

□ KEY POINT — Observation 5.1: Uncertainty Duality

The One-Time Pad achieves total uncertainty on the message axis, but its key axis and message axis are algebraically coupled — $K = M \oplus C$ means knowing the message immediately reveals the key. Enqpy™'s Ideal System achieves two **structurally independent** axes of permanent adversarial uncertainty: Axis 1 (Key) — $H(EK, QK | T^\infty) \geq 2$ bits, permanent key indeterminacy; Axis 2 (Message) — $|S(CT, OR)| \geq 2^{128}$, permanent plaintext indeterminacy. Neither axis is derivable from the other. Knowing the message does not resolve the key, and knowing the key does not resolve the message. No dominance claim is made over OTP; the observation is structural and descriptive. This two-axis independence is a property the One-Time Pad does not realize.

6. Why Quantum Computers Don't Change Anything

Quantum computers can solve certain mathematical problems enormously faster than regular computers. Two of their most famous attacks are called Grover's algorithm (speeds up guessing) and Shor's algorithm (breaks certain math like factoring large numbers).

Both attacks are completely irrelevant to Enqpy™. Here's why:

Grover's Algorithm (Quantum Guessing)

Grover's algorithm speeds up the search for a secret by taking the square root of the search space. If a regular computer needs a billion tries, a quantum computer needs about 31,623 tries instead. But this is only useful when there's ONE correct answer to find. For Enqpy™, there are ALWAYS at least 2 correct answers — both of which satisfy every test equally. You can't speed your way to a unique answer when two answers are permanently identical from every measurable angle. Grover's algorithm finds solutions; it can't make two solutions into one.

Shor's Algorithm (Quantum Math Breaker)

Shor's algorithm breaks ciphers that rely on specific mathematical structures — specifically, the difficulty of factoring huge numbers or solving discrete logarithm problems. Public-key systems like RSA and elliptic-curve cryptography rely on exactly those problems, so they are vulnerable; post-quantum schemes are specifically designed to avoid them. Enqpy™'s security doesn't come from factoring or discrete logs. It comes from the MOD16 16-pairs property, which has no algebraic periodicity structure for Shor's algorithm to exploit. Shor's algorithm simply has nothing to grab onto.

□ KEY POINT

The proof of quantum resistance is not “we think this is hard for quantum computers.” It is: Grover’s algorithm is irrelevant because there is no unique answer to find. Shor’s algorithm is irrelevant because Enqpy™’s structure provides no periodic algebraic pattern for it to exploit. Both of these statements are provable, not assumed.

7. The Vocabulary Dictionary

Here are the key terms used in the formal proof documents, explained in plain language.

MOD16 arithmetic	Clock arithmetic on a 16-position clock (values 0–15). When you go past 15, you wrap back to 0. The key property: any output value has exactly 16 possible input pairs.
Nibble	A 4-bit number, value 0–15. One position in the key. A 256-bit key has 64 nibbles.
PDAF (Key Machine)	The Enqpy™ function that turns master keys into session keys by running many interconnected MOD16 additions. Many inputs produce the same output — that’s the point.
Preimage	All the possible inputs that produce a given output. For any PDAF output, there are always at least 2 preimages (input key pairs). The proof is about the minimum preimage count.
Equivocation	Shannon’s technical term for “how uncertain the attacker still is about the key, even after seeing lots of ciphertext.” The ideal system requires equivocation that never goes to zero.
Ideal System	Shannon’s definition from 1949: a cipher is ideal if, no matter how much ciphertext the enemy collects, they are never left with a unique solution — multiple keys always remain consistent with all observations.
VK (ValueKey)	One of the two components of the PDAF input. The attacker is trying to find the true VK (which equals the master EK). The proof shows they can never narrow it below 2 candidates.
OK (OffsetKey)	The second component of the PDAF input. It controls the ‘displacement pattern’ — which positions get combined during PDAF.
Dependency Graph	A network diagram showing which key positions depend on each other during PDAF. The cycles in this graph determine how many solutions exist.
Cycle	A loop in the dependency graph — a path that comes back to its starting point. Odd-length cycles are the ones that reduce solution count from 16 to 2.

The +8 Shift	The algebraic reason there are always exactly 2 solutions in the worst case: any key and the same key with 8 added to every nibble (mod 16) produce identical outputs. These two keys are permanently inseparable.
Nonce (eff_or)	A fresh random number generated for each message. Makes every session key different even with the same master keys.
EK and QK	The two master secret keys. Must be different from each other. The cipher is proven secure as long as both are kept secret.
Deterministic Success Metric (S)	A binary adversarial outcome function introduced in the April 2026 DSMV companion note. $S = 1$ if the adversary can uniquely determine the plaintext with certainty; $S = 0$ otherwise. It deliberately collapses the question to: did the adversary succeed? This is distinct from Shannon's probabilistic framework, which measures how much the adversary's odds improve per unit of ciphertext observed. $S = 0$ is only meaningful in conjunction with independently proved equivocation bounds — a "substantive" $S = 0$ requires large proved floors; a "trivial" $S = 0$ (near-zero residual uncertainty) indicates a broken system.
Uncertainty Duality	A structural property of Enqpy™ established in the DSMV companion note (Observation 5.1). Enqpy™ maintains two algebraically independent axes of permanent adversarial uncertainty: a key axis ($H(EK, QK T^\infty) \geq 2$ bits) and a message axis ($ S(CT, OR) \geq 2^{128}$). Neither is derivable from the other. This contrasts with the One-Time Pad, where key and message uncertainty are algebraically coupled via $K = M \oplus C$ — a single effective axis. No claim of superiority over OTP is made; the observation is structural and descriptive.
Adversarial Outcome Equivalence	The formal result (Theorem 3.1, DSMV companion note) that Enqpy™ and Perfect Secrecy share the same $S = 0$ outcome under the Deterministic Success Metric, given Enqpy™'s independently proved equivocation bounds. Also refers to Conjecture 4.1 — the more general (not yet fully proved) claim that any cipher satisfying Shannon's dual equivocation conditions with bounds of sufficient magnitude produces the same $S = 0$ outcome as Perfect Secrecy.

8. The Proof in Sequence: Putting It All Together

Let's trace the complete proof from beginning to end — each step building on the one before.

1. MOD16 has 16 preimages for every output (Lemma 1). Pure math. Checkable by hand. No assumptions needed.
2. The Key Machine (PDAF) combines thousands of MOD16 operations in a structured network. The network has cycles.
3. Odd-length cycles create a constraint that cuts solution count from 16 down to exactly 2 (Theorem 1). The two survivors are always {true key, true key + 8 everywhere}. Even-length cycles don't reduce count at all.

- 4. Every ciphertext the attacker captures is reproduced identically by all four coset keys. Collect a billion of them and all four still match every one — because the +8 relationship holds for every nonce, not just one. The four never separate.
- 5. After about 64 messages, the attacker has squeezed out every last drop of information the ciphertext can give. They are still left with four candidate key pairs. They always will be. The equivocation has stabilized at its 2-bit floor ($\log_2 4$). It will never go lower (Theorem 2).
- 6. Quantum computers cannot resolve this. Grover's algorithm can't find a unique answer when two answers are truly identical. Shor's algorithm can't find a pattern when there isn't one. The bound is quantum-invariant (Corollary 2).
- The same structure guarantees the message is also permanently ambiguous: at least 2^{128} different messages are consistent with any ciphertext (Theorem 3, Plaintext Equivocation — closed: the posterior is uniform over the full consistent set). Both the key and the message are proved permanently underdetermined — on two independent mathematical axes simultaneously. This satisfies Shannon's core criterion: no unique solution — ever.
- A companion note (Deterministic Success-Metric View, April 2026) asks the binary operational question: can the adversary succeed? The answer — $S = 0$, meaning no — holds for both Perfect Secrecy and Enqpy™ under the proved equivocation bounds (Theorem 3.1). Shannon's "weaker" designation for the Ideal System retains its mathematical meaning; it does not change this outcome. A further structural observation: Enqpy™'s two uncertainty axes (key and message) are algebraically independent of each other, while the One-Time Pad's uncertainty lives on a single coupled axis (Observation 5.1, Uncertainty Duality).

✓ SUMMARY

Enqpy™ is unreadable because: 1. MOD16 math provably ensures every output always has multiple possible inputs. 2. The PDAF Key Machine amplifies this: from the ciphertext alone, you can never determine which of the four equivalent master keys went in. 3. This is true after 1 message. It's true after a billion messages. It's true for quantum computers. 4. The same structure guarantees the message is also permanently ambiguous: at least 2^{128} different messages are consistent with any ciphertext. Both the key and the message are proved permanently underdetermined on two independent mathematical axes. 5. It's not a conjecture. It's not an assumption. It's a proof — the same kind of rigorous mathematical proof used to establish that $2 + 2 = 4$. Claude Shannon defined what an absolutely secure cipher would require in 1949. Enqpy™ is the first practical cipher to meet that definition — and the message is proved permanently ambiguous too, on a second, independent axis.

9. What the Proof Does NOT Claim

This is important. The proof is about one specific thing: the attacker can never uniquely determine the master keys, no matter how much ciphertext they collect. But a few things are outside the scope of the proof, and being honest about them matters.

Not Covered by Proof	Why It Matters	What to Do
----------------------	----------------	------------

Message integrity	The proof guarantees confidentiality — the contents stay secret. It does not guarantee that a tamperer couldn't modify a ciphertext without being detected.	Add a MAC (Message Authentication Code) to every message, as specified in the implementation guide.
Side-channel attacks	If an attacker has physical access to the device running Enqpy™ and can measure power consumption or timing, they might gain information about one session key. This is true of every cipher.	Secure the device. A session-key exposure in one message does not reveal the master keys, and every other message stays protected by its own fresh nonce.
Key management	If you write your master key on a sticky note on your monitor, no cipher can protect you. Key generation, storage, and distribution are outside the cryptographic proof.	Follow the key entropy and CSPRNG requirements in the specification.
Chosen-key attacks	The proof assumes EK and QK are genuinely independent random keys. If they were identical or had very low entropy, the security properties weaken.	Always use a proper random number generator. Never use EK = QK.
Known plaintext + a reused key	The whole proof is ciphertext-only: it assumes the attacker sees ciphertext and never the matching plaintext. If the attacker already knows the plaintext behind some messages and the same key is reused, the keystream's MOD16 structure becomes solvable — the working key is recovered (up to the +8 coset) and later messages under that key can be read.	Deploy with a fresh, independent key per message, as the implementation guide specifies. A known-plaintext exposure in one message then can't transfer to any other.

⚠ REMEMBER

The proof gives Enqpy™ a security guarantee that no other practical cipher in history has had: the master keys are permanently ambiguous regardless of computation. But like any cipher, Enqpy™ requires correct implementation, proper key generation, and integrity mechanisms for complete security. The proof is not magic — it's a specific, bounded, formally rigorous guarantee about one specific and very important property.

10. One Last Analogy: The Librarian's Code Book

Here's one final way to think about the whole proof.

Imagine a librarian who has encoded a secret message. The librarian used one of two special code books — Book A or Book B. The books are different, but they produce IDENTICAL ciphertexts for

every possible message. The librarian chose one and encoded the message. You see only the ciphertext.

You can spend your entire life collecting more encoded messages from this librarian. You can observe every message ever sent. But every single message is equally consistent with Book A and Book B. You have seen everything you can possibly see, and you are exactly as uncertain about which book was used as you were before you started.

Why? Because Book A and Book B are not just similar — they are mathematically, provably, permanently equivalent at the level of what they produce. The question “which book did the librarian use?” has no answer that can be recovered from what the attacker sees. The information required to answer it is structurally absent from every ciphertext.

That’s Enqpy™. Book A is the true master key. Book B is the true master key with 8 added to every nibble. Both produce the same encrypted output for every message, every nonce, every session. The distinction has been provably removed from the ciphertext by the mathematics of MOD16 and the PDAF Key Machine.

□ KEY POINT

Shannon called this property “key equivocation that never vanishes.” He proved in 1949 that it was theoretically possible for finite-key ciphers. Enqpy™, 77 years later, is the first practical cipher to actually achieve it — on both axes simultaneously: the key is permanently ambiguous (Section 5.2), and the message is permanently ambiguous (Section 5.4). The librarian’s secret is kept on both fronts: you cannot determine which code book was used, and you cannot determine what the original message said.

About This Document

Field	Value
Title	How Enqpy™ Proves It Can't Be Read — A Plain Language Guide
Source Documents	Enqpy™ Formal Cryptographic Description Rev 5.0; Enqpy™ Formal Proof Paper Rev 5.0 (incorporating the Formal Information-Theoretic Proof, Open Items Closure, and Plaintext Equivocation Theorem, with revisions through Rev 5.0 noted in the paper); A Deterministic Success-Metric View of Shannon’s “Weaker” Designation for the Ideal System (NQP LLC Technical Note, rev. June 2026)
Intended Audience	General readers with no cryptography background; junior high school level and above
Cryptographic Claims	All claims in this document are derived directly from the formal proof documents listed above. No claims have been simplified in a way that alters their meaning.
Author	Paul McGough / NQP LLC, Manassas, Virginia
Year	2026